

**APPARATUS AND METHOD OF UPGRADING PROGRAM OF FIRMWARE****BOARD****CLAIM OF PRIORITY**

5           This application makes reference to, incorporates the same herein, and claims all  
benefits accruing under 35 U.S.C. Section 119 from an application for PROGRAM  
UPGRADING APPARATUS AND METHOD FOR FIRMWARE BOARD filed earlier in  
the Korean Industrial Property Office on June 9, 1999 and there duly assigned Serial No.  
21353/1999.

**BACKGROUND OF THE INVENTION****1. Field of the Invention**

10           The present invention relates generally to a firmware board having an incorporated  
function program stored in a non-volatile memory. More particularly, the present invention  
15       relates to an apparatus and method for upgrading the function program of the firmware board.

**2. Description of the Related Art**

          Typically, a firmware board is provided with an erasable and programmable read only  
memory (EPROM), an electrical erasable and programmable read only memory (EEPROM),  
and other non-volatile memory to store pre-programmed boost-up data (i.e. firmware) relevant

to establish information or configuration therein. The EPROM, which stores the program, is manufactured using an EPROM writer, then the manufactured EPROM is superimposed on the nonvolatile memory of the firmware board to drive the firmware board. As occasion demands, the program for driving the firmware board is upgraded.

However, the prior art system of upgrading the program for driving the firmware board has drawbacks. The first drawback is that it requires additional hardware, i.e., the EPROM writer itself. The second drawback is that it is inefficient and inconvenient as the EPROM already packaged on the firmware board must be physically removed from the backbone while the power of the firmware board is turned off to replace the old EPROM with a newly manufactured EPROM.

006030" 5421650

## SUMMARY OF THE INVENTION

Accordingly, the present invention has been made in an effort to solve the problems occurring in the related art, and therefore, an object of the present invention is to provide an apparatus and method for upgrading the program for driving a firmware board without using a  
5 separate tool, such as the EPROM writer.

It is another object of the present invention to provide an apparatus and method for upgrading the program for driving a firmware board that can eliminate the intervention to the hardware by the end user when upgrading the program.

10 It is still another object of the present invention to provide an apparatus and method for upgrading the program for driving a firmware board so that the program can be more easily upgraded by simply downloading new firmware data to acquire new capabilities.

In order to achieve the above objects in the embodiments of the present invention, the program of the firmware board can be upgraded by the built-in program and protocol without  
15 turning off the power of the firmware board and without using a separate hardware tool, such as an EPROM writer.

According to the present invention, a flash memory is provided on a firmware board to store the program for production. A host computer creates a file for production by converting

an execution file, prepared in advance, into the file for production. A personal computer (PC) receives the production file from the host computer by downloading therefrom, and the PC stores the production file in the corresponding region of the flash memory using a production-processing program of the firmware board.

006030" 54216560

## BRIEF DESCRIPTION OF THE DRAWINGS

The above objects and advantages of the present invention will become more apparent by describing in detail the preferred embodiment thereof with reference to the attached drawings in which:

5        FIG. 1 is a block diagram illustrating the construction of an apparatus for upgrading the program for driving the firmware board according to the present invention;

FIG. 2 is a flowchart illustrating a file creating process according to the present invention;

FIG. 3 is a view explaining the structure of an .INI data file used for creating the file  
10        for production according to the present invention;

FIG. 4 is a view illustrating the structure of the file for production that is created according to the present invention;

FIG. 5 is a view illustrating the structure of a header of the file for production created according to the present invention; and,

15        FIG. 6 is a flowchart illustrating the process of transmitting the file for production onto the firmware board and for storing the file for production in the corresponding region of a flash memory according to the present invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In the following description, for purposes of explanation rather than limitation, specific details are set forth such as the particular architecture, interfaces, techniques, etc., in order to provide a thorough understanding of the present invention. However, it will be apparent to those skilled in the art that the present invention may be practiced in other embodiments that depart from these specific details. For the purpose of clarity, detailed descriptions of well-known devices, circuits, and methods are omitted so as not to obscure the description of the present invention with unnecessary detail.

FIG. 1 depicts a block diagram illustrating the construction of an apparatus according to the present invention for upgrading the program for driving the firmware board. The upgrading apparatus employs a flash memory 124 in which a data read/write operation is performed utilizing software for storing the program with new capabilities onto a firmware board 120, instead of replacing the EPROM as in the prior art.

With reference to FIG. 1, a host computer 100, which operates under the control of an operating system, converts an execution file prepared in advance by a programmer into a file for production by converting the object codes into an appropriate format to be stored in the firmware board. Here, the host computer 100 can be a UNIX system. The execution file includes program object codes for upgrading the program that are used for driving the firmware board. The host computer 100 creates the file for production using a software

known as a "SIGX2B" execution program to convert the program object codes into the file for production.

It should be noted that within this disclosure, the term personal computer (PC) is meant broadly to refer to any type of programmable terminal – i.e., notebook PC, cellular phones, digital cameras, or other small-size information apparatus.

A personal computer (PC) 110 downloads the production file created by the host computer 100 and stores the production file in a corresponding region of the flash memory 124 of the firmware board 120. In order to actuate the transfer, the PC 110 is provided with a communication (COM) port 112 for loading the production file to the firmware board 120. The communication port 112 is connected to the input/output port 122 of the firmware board 120 through an RS-232C line. Here, the terminal emulator of the PC 110 performs the transmission of the file through the communication port 112.

The firmware board 120 is provided with a non-volatile memory, flash memory 124, in which the data read/write is possible using software and stores the drive program therein. That is, the flash memory 124 stores the production-processing program, which is stored initially in the flash memory 124 using an EPROM writer, and thereafter, the program production/upgrading is performed in a manner that any program is stored in the flash memory 124 using the production-processing program. The production-processing program receives the file which was converted to the production format using the "SIGX2B" program

from the PC 110 and saves it onto the flash memory 124. Then, after upgrading with the new program, the boost-up program boots this new file stored in the flash memory 124. Accordingly, the PC 110 can upgrade the program stored in the flash memory 124 using the production-processing program.

5

Moreover, in the embodiment of the present invention, the firmware board 120 includes a Dynamic Random Access Memory (DRAM) 126. If the file for production created by the host computer 100 corresponds to the file for upgrading the production-processing program, the production process is performed by copying the production-processing program from the flash-memory 124 onto the DRAM to perform the upgrading process therein, then the upgraded production-processing program is stored back into to the flash memory 124 again. For upgrading the operation, the embodiment is provided the DRAM 126. It should be noted that a Static Random Access Memory (SDRAM) can be also used.

FIG. 2 is a flowchart illustrating a process of creating a file for a production performed by the host computer 100 according to the present invention. The following will be a general discussion of the present invention in the context of a software related to the PC and the firmware. It is to be understood that the software and hardware can take many different configurations such as is within the skill of those skilled in the art.

With reference to FIG. 2, the host computer 100 creates an execution file by compiling/linking operating programs, MODULE 1, HEX,.....,MODULE n, HEX, for

5    respective modules in step 210. Then, the host computer 100 creates database files, DATA 1, BIN, ....., DATA n, BIN, in step 220. Thereafter, in step 230, the host computer 100 converts the respective execution files created in step 220 into files for production, PGM 1, 000, ....., PGM n, 000, using a "SIGX2B.EXE", which is a tool for creating the files for production. At this time, the files for production are created after a file of ".ITC.INI" is executed. However, the ITC.INI is a normal test file which can be produced using any editor program and is created before the production file is made. The ".ITC.INI" is an input data file for "SIGX2B.EXE" and notifies the "SIGX2B.EXE" with the configuration and method for making a production file. Thus, the ".ITC.INI" file is a file which clarifies the production
 10    guide to the production file and includes information on storage regions, booting, etc., of the flash memory 124. This information is contained in the header portion of the production file. Input parameters can be set by an operator using a single, .INI file to specify application parameters and defines the starting parameters for the system. That is, the .INI file specifies the application, services, driver letter, program, install file, set-up program, memory space,
 15    user name, and/or message type. In step 240, the file for production created in step 230 is downloaded to the PC 110.

As described above, the "SIGX2B" execution file is a tool for creating S-record files and binary files as production files. When creating the production files, it follows instructions recorded in the .INI file. Thus, by executing the "SIGX2B package name .INI", the files for
 20    production are created.

FIG. 3 is a view explaining the structure of the .INI data file used for creating the production file according to the present invention. This .INI data file is an input text file of the "SIGX2B" execution file. A list of files which constitute a package for production and commands for the package construction are recorded in the .INI data file.

#### INI File Parameter

1. Reference number, 300, denotes a region where information related to the firmware board is stored.
2. Reference number, 301, denotes an information tag related to the firmware board.
3. Reference number, 302, denotes a package identifier of a fixed value, i.e., the name of the firmware board.
4. Reference numbers, 310, 320, and 330, denote regions where information related to respective modules/components of the firmware board is stored.

As information related to the respective modules can be stored in the same manner, only the first module 310 will be explained as an illustrative example hereinafter.

5. The reference number, 311, denotes an information tag related to a module (MODULE).
6. The reference number, 312, denotes a file name of modules (File Name) which constitutes a package.

7. The reference number, 313, denotes the booting state of the file (LoadFlag). For example, "0" represents that the file has not been booted and "1" represents that the file has been booted.
8. The reference number, 314, represents the type of file stored in the flash memory 124. For example, "0" represents that the file is of a non-compressed type and "1" represents that the file is of a compressed type.
9. The reference number, 315, represents the type of an input file (BINflag). For example, "0" represents an S-record file and "1" represents a binary file.
10. The reference number, 316, represents the position (unit of 64Kbytes) of the flash memory 124 where the file is stored (SaveAddr).
11. The reference number, 317, represents the maximum size of a resultant file (SaveSize). If the resultant file exceeds 64Kbytes, it is created as "\*.000, \*.001".

FIG. 4 is a view illustrating the structure of the production file created according to the present invention. The production file is created through the process as shown in FIG. 2.

With reference to FIG. 4, the file for production comprises a file header 410, and file content 420. The file header 410 is composed of 32 bytes, and the file content 420 is composed of 64Kbytes – 32bytes. The production file can have the size of 128Kbytes at maximum. The file content 420 can be a compressed type according to ZipFlag of the .INI file.

FIG. 5 is a view illustrating the structure of the header of the production file created according to the present invention (i.e., the structure of the file header 410 of FIG. 4).

With reference to FIG. 5, the header 410 of the file for production comprises “ID (00h~03h)”, “ATTRIB (03h~04h)”, “FILE NAME (04h~0Fh)”, “BOOTING ADDRESS (10h~14h)”, “BOOTING SIZE (14h~18h)”, “SAVE ADDRESS (18h~1Ch)”, and “SAVE SIZE (1Ch~1Fh)”.

#### Header Parameters

1. The “ID” represents the package identifier and indicates the ID of the .INI file.
2. The “ATTRIB” represents the attributes of the file that exist in the .INI file. At this time, “bit 0” represents the booting state and “bit 1” represents the compression state.
3. The “FILE NAME” represents the file name of a module that exists in the .INI file.
4. The “BOOTING ADDRESS” represents the position where the corresponding file is booted and extracted from the execution file.
5. The “BOOTING SIZE” represents the booting size of the corresponding file that is extracted from the execution file.
6. The “SAVE ADDRESS” represents the position of the flash memory 124 where the content of the corresponding file is stored.
7. The “SAVE SIZE” represents the size of the content of the corresponding file.

FIG. 6 is a flowchart illustrating the process of transmitting the production file created according to the present invention to the firmware board and for storing the production file in the corresponding region of the flash memory. This process is performed by the terminal emulator of the PC and the production-processing program of the firmware board 120 using the production file that is downloaded from the host computer 100.

If an operator inputs a command through the PC 110 to transmit the file for production, i.e., if the operator enters a return key on the terminal emulator of the PC 110 (step 610), the PC 110 connects a communication path to the production-processing program of the firmware board 120 through the RS-232C communication line using the terminal emulator. In response to the input of the return key, the production-processing program outputs a command input prompt to display "ITCMON>" on a display screen of the PC 110 (step 620). In response to the display of "ITCMON>" on the display screen of the PC 110, the user may enter a SIGLO command, i.e., a command for transmitting the file for production. If so, "ITCMON>SIGLO" is displayed on the screen of the PC 110 (step 630). The PC 110, in response to the input of the command for transmitting the file for production, transmits the production file to the firmware board 120 (step 640).

The production-processing program of the firmware board 120 receives the production file transmitted from the PC 110 and analyzes the header information of the received file for production (step 650), then stores the production file in the corresponding region of the flash memory 124 according to the analysis result (step 660). At this time, if the region of the flash

memory 124 where the production file is to be stored is the region where the production-processing program is stored - i.e., if the production-processing program itself is to be updated - the production-processing program copies its own program to the DRAM 126, jumps to the DRAM 126, then performs the production-process on the DRAM 126 (step 670). That is, the updating of the production-processing program is performed in the DRAM 126. If the updating of the production-processing program is completed, the updated production-processing program is stored back in the corresponding region of the flash memory 124 (step 680).

As described above, according to the present invention, a firmware program is upgraded by a production-processing program stored in a firmware board, and thus a separate hardware tool such as an EPROM writer is not required. In other words, the program of the firmware board can be upgraded only by the replacement of an operating program in the flash memory and not by the hardwired replacement of the EPROM. Thus, the program of the firmware board can be upgraded without turning off the power supply to the firmware board. Moreover, as the upgrading of the firmware program can be achieved independently from an upper system where the firmware board is packaged and the upgrading is performed only for the module that requires the upgrading, the inconvenience of replacing the EPROM, which stores the programs of the whole firmware board, can be prevented.

Further, the present invention has advantages in that module execution files can be freely positioned in the desired region of the flash memory using the tool (SIGX2B) by

converting the execution files prepared by a programmer into the production files using the .INI file for specifying the configuration of production. Also, the present invention have advantage of using the firmware board without regards to the operation type of the firmware board since the booting status and the booting address can be designated in the event that the  
5 firmware board operates not only directly on the flash memory but also through booting the DRAM.

The previous description of the preferred embodiments is provided to enable any person skilled in the art to make or use the present invention. The various modifications to these embodiments will be readily apparent to those skilled in the art. Thus, the present  
10 invention is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein. Therefore, it is intended that the present invention not be limited to the particular embodiment disclosed as the best mode contemplated for carrying out the present invention, but that the present invention includes all embodiments falling within the scope of the appended claims.

15

20